ANIMATING A COST EFFECTIVE CHARACTER FOR AN EDUCATIONAL PRODUCTION

A Thesis

by

LUKE ANTHONY CARNEVALE

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2004

Major Subject: Visualization Sciences

ANIMATING A COST EFFECTIVE CHARACTER FOR AN EDUCATIONAL PRODUCTION

A Thesis

by

LUKE ANTHONY CARNEVALE

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Donald H. House (Chair of Committee)	Frederic I. Parke (Member)
Richard E. Orville (Member)	Phillip J. Tabb (Head of Department)

August 2004

Major Subject: Visualization Sciences

ABSTRACT

Animating a Cost Effective Character for an

Educational Production. (August 2004)

Luke Anthony Carnevale, B.E.D., Texas A&M University

Chair of Advisory Committee: Dr. Donald H. House

Animation is a powerful teaching tool. Ideas can be expressed through animation using only a fraction of the time needed with conventional teaching methods [John Halas 1987]. In short, a picture is worth a thousand words. However, educational budgets rarely allow for the expensive time-consuming task that animation entails. This thesis tackles the challenge of creating substantial quality educational animation using limited time, labor and money resources. A National Science Foundation sponsored planetarium show on lightning will be used as a demonstration project to document the techniques employed and results achieved. Anyone interested in reducing costs can reference this thesis for insight on what or what not to implement in their own production process.

ACKNOWLEDGEMENTS

I would first like to thank God, who kept me striving towards what, at times, seemed like an ever distant finish line.

Much thanks to Dr. House who has been an invaluable resource, guide, and friend throughout. I would also like to thank my committee members, Dr. Parke and Dr. Orville, for the incredible help their knowledge and direction has been.

Thanks to Chris Haaser and Jennifer Kuhnel who have helped me get going and stay going in the right direction; and to the entire Enlightening Lightning staff for their great input and for making my thesis so enjoyable.

Thanks to Dr. Kristina Winegarden for her constant understanding and support. Finally, thanks to my family for being the greatest family ever!

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

Animation has been used for centuries to teach and to tell stories. From the ancient animation of wooden puppets by a skilled puppeteer, to modern photorealistic computer generated scenes, animation has captured the minds and hearts of adults and children alike. Many educational productions have recognized the value of this approach. "With animation, ideas can be expressed using only a fraction of the time needed with conventional teaching methods [6]." It is a great way to educate and entertain audiences. Children's shows such as Sesame Street and the Muppets have capitalized on the use of hand animated puppets and simple animations. Others have opted for a more traditional approach of clay or 2D animations, while still others have chosen to explore modern 3D technologies. The media may be quite different, but the idea behind them is the same. The animated story is a powerful story. Thus, it is a powerful tool for teaching and educating.

Understandably, production companies want to be able to use such a compelling technique to tell their stories. However, unlike the Muppets, many production companies, particularly educational companies, are not afforded the luxurious budgets of large studios. It is much harder for these companies to produce quality 3D animations, which require substantial time and money resources.

This thesis follows the style and format of *IEEE Transactions on Visualization and Computer Graphics.*

Fortunately, anyone who attempts computer animation soon realizes it is not necessarily physically correct motion or scientifically accurate simulation that makes good animation. It is what *looks* physically correct. In other words, if it looks right, it is right. Hence, creating the optimal balance of computer workload, between physically correct and looking right, is the ultimate goal of any production. This is commonly called *automating* and *streamlining* in computer animation. Automating and streamlining are really just technical names for "cheating" by making intelligent choices, excluding extraneous details and only using what truly communicates. Namely, in this case, it is the cheating of time costs. Because, as Blinn so eloquently titled his 1988 Siggraph paper [1], computer animation is *The Ancient Chinese Art of Chi-Ting*.

This thesis explores the problem of producing large amounts of quality computer generated animation for productions where both money and labor resources are quite limited. The demonstration project for this thesis stemmed from a grant through the National Science Foundation (NSF) for the creation of a full length, forty minute planetarium show on lightning. While a common production process was followed for this project, care was taken to record the specific strategies used to complete production at low cost of both time and labor.

The thesis describes the various techniques and tricks implemented to save production costs. It also includes an informal evaluation of the full production process. Combined, these may be used as a set of loose guidelines for interested parties of what to do and what not to do. Thus, anyone interested in stretching their own animated production's budget may reference this thesis to increase and achieve more efficient results.

The goal of this demonstration project was to create a fully animated narrator character that would handle ten to twelve minutes of dialogue, half of that time being on screen. This was done using a budget only large enough for one person to provide the total labor resources from concept to completion. Close to eleven thousand frames were modeled, rigged, animated, lit, textured, and rendered under these limitations. This meant that the character had to be cleverly designed to minimize as much production cost as possible. The idea was to ease the animator's load by giving the computer as much of the work as possible, without unnecessarily slowing down the computer. Numerous tricks and shortcuts had to be created to "cut" as many corners as possible.

CHAPTER II

BACKGROUND

It would be misleading to suggest that animation has one unique origin [6]. Since the beginning of time, humans have tried to capture a sense of motion in their art [7]. Animation itself has been "invented" so many times by so many individuals that its true starting point could be placed almost anywhere [6]. From the the Altamira caves of Northern Spain to the scrolls of ancient Japan [7], remnants of early animations can readily be found. "…This quest for capturing motion has been a common theme throughout many on mankind's artistic endeavors [7]."

Animation cannot be truly understood without first explaining a fundamental principle of the human eye. This principle called *persistence of vision*, describes a natural phenomenon which occurs in the human visual system. Any image viewed by the eye is retained by the retina for a brief instant, even after the image has been removed from view. This basic principle is what allows a series of still pictures presented in a rapid sequence to blend together into one continuous image. Frenchman, John Paris, popularized this concept with his 1825 invention of the *thaumatrope*. As Halas [6] described, this device was a circular disk with a picture of a bird on one side and a birdcage on the other. When the disc was rapidly twirled by means of strings, the bird appeared to be in the cage. This was a simple display of the power of persistence of vision. Throughout the late 1800's other inventions like the *phenakistoscope* and *zoetrope* slowly took steps in the direction of animation as we know it today. However, the first real practical means of making animation wasn't until Thomas Edison's development of the motion camera and projector [7].

The first animation production house, opened in 1913 by Raoul Barre', marked the commercial birth of the animation industry. Within five years the industry was well on its way, as more and more studios began to pop up. As Culhane [4] states, Winsor McCay, rightly called the father of the animated cartoon, pioneered the way. Culhane continues, the producers of the day all thought of animated cartoons as simply a moving comic strip. The drawings had no weight, sag, or squash. McCay's 1914 ground breaking animated cartoon, *Gertie the Trained Dinosaur,* displayed animation abilities not equaled until Disney began to improve his own product in the 1930's. The reason that McCay's work was so important was that, early on, he grasped the possibility of making animation appear lifelike. His characters were animated with weight, maintained constant volume, moved in perspective, and even had overlapping actions. Unfortunately, other animators of the time disregarded McCay's lead, and great animation was forced to wait for Disney's touch.

As Culhane [3] explains it, the initial animation process began with the head animator drawing the key movements of a given action, numbering and annotating the drawings on an exposure sheet. The work then goes to an assistant animator, who follows the exposure sheet instructions and adds more drawings. This work is then passed on to a second assistant to finish the final in-between drawings. The work is all done on an animators drawing table. In the center of the table is a glass pane and under it is a light box. The paper used is thin enough so that, when the light is on, the animator can see the animation drawings through four or five sheets of paper. This allows the relative positions of several animation drawings to be easily gauged.

Assuming the work is approved, continues Culhane, the drawings were transferred to colorless *acetates*, called "cels." These cels were then colored using opaque paint. Because the cels were transparent, multiple layers could be used. Backgrounds and movements could then be separated into components, allowing for the minimal amount of drawing work to be done. Finally, these completed cels were collected, photographed one exposure at a time on an animation stand and combined on film. The film was finally combined with the sound track to create the finished product.

"Between the late 1920's and the late 1930's animation grew from a novelty to an art form at the Walt Disney Studio [9]." "Walt Disney was not so much an animator as a phenomenon of the twentieth century [6]." With innovations like sound effects and music added to his cartoons, Disney was constantly in the animation forefront. New techniques and ideas were continuously tested.

As Halas [6] writes, Disney invested far more time in storyboarding than did any other studio. Enormous amounts of consideration went into this initial stage of production and the value of this policy invariably paid off. As a result many of the creative problems were solved before animation was even begun.

In 1937, the release of *Snow White and the Seven Dwarfs* set a precedent for all animated films to follow. Over the subsequent years Disney introduced innovative ideas such as rotoscoping, staging, timing, and many other principles of animation which are followed to this day. He was a key pioneer in developing set rules, guidelines and "tricks" to aid in the speed, quality and believability of animation.

William Hanna and Joseph Barbera were pioneers in their own right. After MGM's 1957 decision to close shop, they established their own production house for

television cartoons under the name Hanna-Barbera. Stabile [13] explains, these cartoons were based on the idea of "limited animation" – essentially the reduction of animation to its most basic form: little motion, no complex choreography, repeated cycles of movement, a small repertoire of expressions and gestures, stress on dialogue, basic design, and simple graphic forms. The simplistic style Hanna-Barbera employed in their early television cartoons was largely due to financial constraint. Nevertheless, their work was still made with the kind of ingenuity that embraced the fundamental principles of pioneering animators in the field. Stabile continues, the focus on a smarter dialogue aimed at adults and visuals for the kids allowed for a much broader audience base. This new audience was appropriately termed, "kidult." Children liked the shows because of the action and animation, and adults enjoyed what they heard. While others sought to brand this new style "illustrated radio," Hanna-Barbera shows like *The Flintstones* paved the way for modern shows such as *King of the Hill* and *The Simpsons*.

With the introduction of computers, animation began to change. The capability of calculating and storing large amounts of three dimensional data was becoming a reality. Initially traditional artists were hesitant to jump on this new technology. The word *computer* sent a xenophobic chill through creative workers in cel animation. There was a strong suspicion that computers would take control of the creative process, or even replace it [3]. Yet, as programmers produced more powerful and user friendly software, artists slowly began to see its amazing capabilities. John Lasseter was one such early artist. He greatly advanced early computer animation with his work at the first exclusively computer graphics (CG) studio, Pixar. He was among the first to describe "the basic principles of traditional 2D hand drawn animation and their application to 3D

computer animation [11]." He understood that the end result of CG animation was a sequence of two dimensional images, as it had always been. It was just the process of getting there which had evolved. With his knowledge of the traditional aspects of 2D animation and the newly created CG technology he wrote, *Principles of Traditional Animation Applied to 3D Computer Animation* [9]. In it Lasseter described eleven traditional principles and how they applied or adapted to this new technology. Essentially, all the principles Disney had created years before still had to be represented in the CG final product. The computer did not replace or even change the fundamental animating principles, but rather it was just a new tool for creating them.

With award winning shorts like *Luxo Jr.* and *Red's Dream* [11], Pixar began to open the eyes of the public and other artists alike. They began to realize the vast possibilities and wondrous capabilities of this newly harnessed technology called computer graphics. After other noteworthy shorts and numerous television commercials, 1995 saw Pixar premier its first ever full length feature film, *Toy Story*. It became the highest grossing film of the year drawing in a total of $365 million world wide [11], and the rest, as they say, is history!

CHAPTER III

METHODOLOGY

The following is a step-by-step discussion of the methods used in the production of character animation for the planetarium show project.

III.1. General Problems

Animating an involved character generates many significant problems, both foreseen and unforeseen. Some, case specific, must be solved as they arise, but there are quite a few general problems that can be prepared for. All the non-planetarium specific problems that must be faced can all be conveniently lumped into the same category. They all boil down to just one thing, complexity. The more detailed a character, the more realistic it must look and move to make an audience accept it. Often, high levels of complexity are unnecessary. "Characters with simple shapes are often more effective than complex ones [10]." Many of the same emotions can be displayed by a ball in a box, as by a dragon in an ancient castle, but with a lot less work.

There are several steps in the overall production process of an animated character. First the character must be designed, *modeled* and built in a three dimensional environment, where it will be stored as a large set of 3D points in the computer. A system for movement, or *rig*, is then added to the character. The rig works much as a human skeleton works. It allows the animator to more easily produce the final desired movements. Much of the streamlining and automation must be developed and implemented in this stage of the process. Next, *light* and *texture* are added to the character. This provides the color and surface texture information used in the final

renderings, or calculated 2D image sequences. The *animation* process, giving the character motion or life, commonly falls between the rigging and lighting stages in the production. However, to minimize lighting and texturing time, the animation for this project was done last. While animation is one of the more time consuming steps in the production, it is only a small part in the overall process. Finally, rendered 2D image sequences are calculated. These sequences are what is finally shown to the viewing audience.

Character choices made at the start of the production will affect the entire process. The more complex the character the more modeling, rigging, lighting, texturing, and rendering must eventually be faced. Not only does more complexity mean more work in number of objects, but it means more realistic texturing, lighting, and animation will be needed to maintain audience believability. Increased complexity also creates a need for more complicated rigs. A rig must be designed to handle any part of the character or object that could move. Therefore, any extra features on a character, no matter how small or seemingly unimportant must still be incorporated into the control design of the rig. Once all of the above is completed, final render times for such a complex character can be substantial. The same can be said for the surrounding scene and props used in aiding the character's performance. Typically, an increase in complexity will also increase the amount of work and time needed to go from start to finished product.

Another problem, specific to the problem at hand, is the environment in which the final product is to be shown. Since it is being created specifically for use in planetarium shows there are a few other factors to consider. The character needs to interact not only with a viewing audience but also with images being displayed simultaneously around the

dome. Therefore, a constant awareness of what is happening around the dome must be maintained, so that the character's performance is always appropriate. Occasionally, planetarium specific equipment like the *DIGI STAR II*, a vector based star-displaying device, will be used in tandem. Often, these machines have special needs. The DIGI STAR II, for example, requires low levels of light to be seen properly. The design of the character must take these needs into consideration so that the final visual results will be viewed effectively.

III.2. Conception and Design

The first step was to address the main character. When thinking about a character design, *appeal* and *ability to convey* the story take precedence. As Wrabel [14] states, "appeal is very important from the start. [It is] anything that a person likes to see." A picture that is too complicated or hard to read lacks appeal [9]. The final character design should embody a complementary balance between both appeal and effective story telling. The initial character design of this project was to be based on the idea that the character was to be a single electron, lost from his molecule "family". As detailed in Fig. 1, the character, Sparky, was designed with simplicity and audience appeal in mind. The design was intentionally kept as simple as possible so that no more time than was absolutely necessary would be spent in the production process. Several characteristics, such as antenna, simple body shape and cute facial features were guiding principles, while variations on these ideas and others were explored. Through the use of design sketches it was determined that a set of hands would also be needed for Sparky to convey the story in a more effective manner.

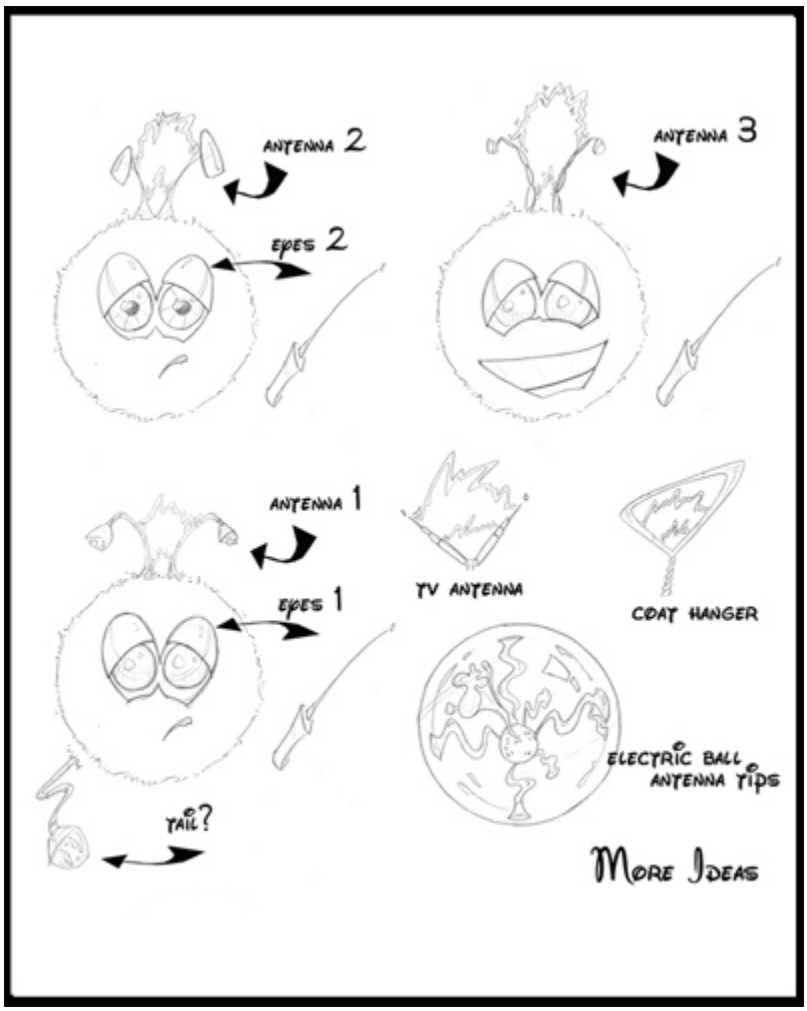


Fig. 1. Preliminary design sketches of "Sparky" character.

The resulting character was a simple spherical body with floating head, hands and automated secondary features. This design eliminated the use of any extraneous, time-wasting appendages while ensuring that the character had all it needed to communicate successfully. In addition, the simple design was intended to facilitate character acceptance and believability by the viewing audience. "An audience will accept any convention, any point of view, as long as it is carried out consistently [3]." Cartoon-like motions are more acceptable because they are much easier to consistently animate then

realistic motions. Realistic motion, like that of a human, requires subtle changes throughout the character's entire body as the overall motion changes. This is extremely difficult through CG key-frame animation and is almost always done using *motion capture*, a system of sensors recording motion data from live actors. This technique however, was beyond the scope of this project. Not only is realistic motion more complex but it invites the highly critical eye of the viewing audience, an audience that knows from everyday experience exactly what human motion should look like.

III.3. Modeling

The character and props were modeled using polygonal and subdivision surfaces [8]. Rather then building a single complex model, each model was designed using separate intersecting pieces of geometry. These pieces of geometry, kept simple, smooth and elegant, were then combined to create the illusion of a single complex model. Props like the small remote in Fig. 2 were modeled in this fashion. By using this approach, the need for UV texture mapping and complex shaders was minimized. In addition, many of the simple geometry pieces could be slightly modified and then reused. This allowed the models to be constructed quite quickly. The exploded view in Fig. 2 further illustrates the modeling technique and how it resulted in the desired cartoon-like style.

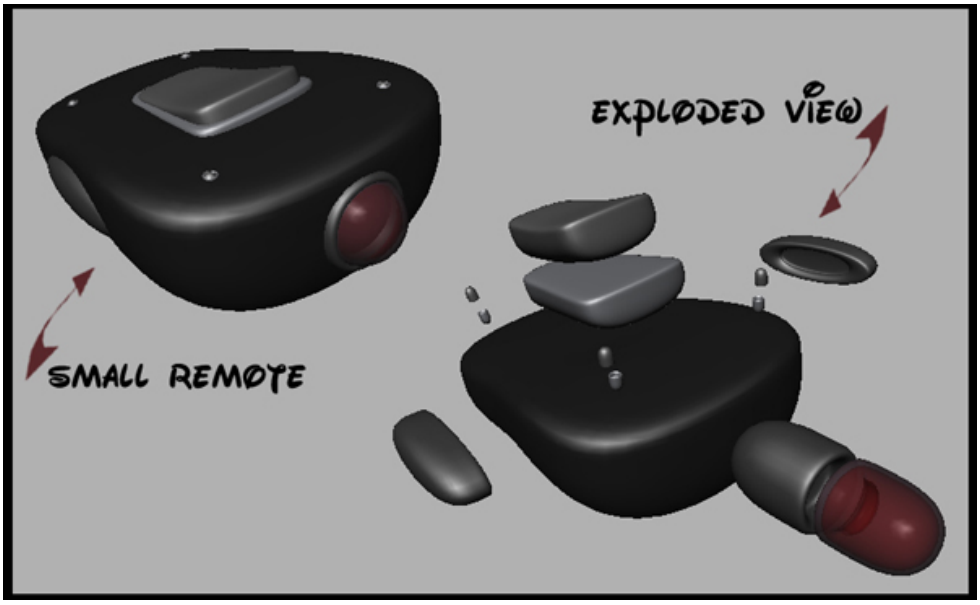


Fig. 2. Remote detail and exploded view.

III.4. Rigging

A *rig* is a system created to control the movements and deformations of the objects or characters in a scene. The goal is to create the most effortless, user friendly system possible. Working with the rig should be an intuitive process. It should be easy for the animator to select and operate controls. A completed rig works much like the human skeletal and muscular systems, telling previously attached parts of the object's geometry how, when, and where to move.

Achieving this commonly begins with the use of *joints*. Joints, like the examples in Fig. 3, are the basic animation tool used for abstracting, or simplifying, the desired movements of geometry. A joint simply defines a location and axis in space about which movement will occur. Computer generated joints work much like human joints and can be *parented*, or placed in a hierarchical order, creating "bones" between them. These groups of appropriately parented joints and bones comprise the rig's skeleton. Additional

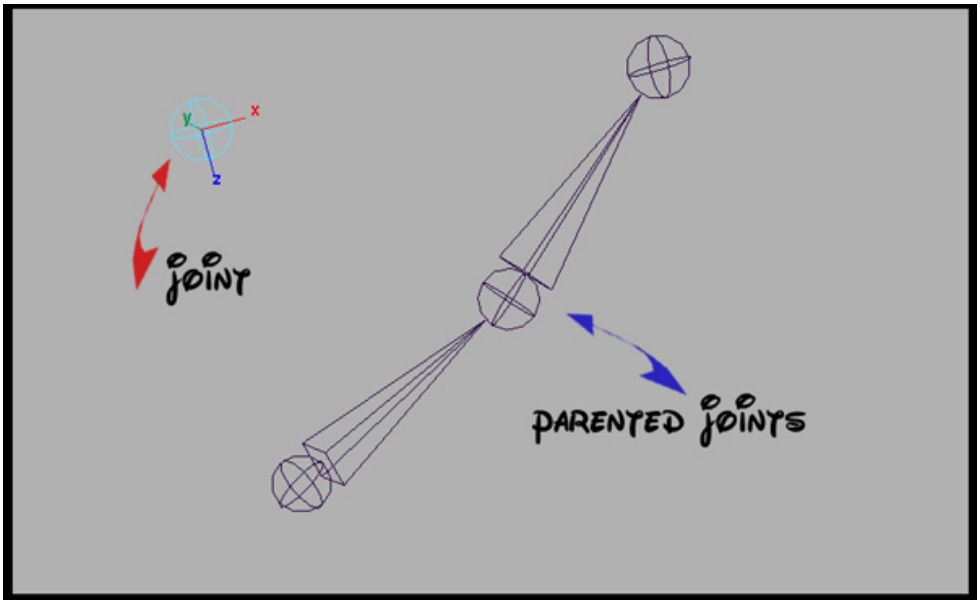


Fig. 3. Joint detail.

systems for movement are then placed on the skeleton. These systems are used to make the animator's job simpler and faster when manipulating the character.

A final level of user controls is often added to the rig, simplifying use one step farther. These controls are commonly created from some form of simple geometry and are constrained to the rig. They become like "handles" for the animator to select and manipulate. The idea behind a user control is that the animator will never have to deal directly with the rig itself. Instead, the animator moves the rig by selecting and manipulating a control "handle" or its attributes. Each control has previously assigned attributes. These attributes are connected to joints or groups of joints performing some movement. Desired motion can be created by simply selecting a control and performing operations on its attributes.

Fig 4 was included as a visual summary of a rig hierarchy. The very base level begins with the vertices of the geometry to be animated in the scene. The combined movement of these vertices is what the viewer will see as the final animation. The next level up in the hierarchy is the skeleton comprised of bones and joints. Each vertex is assigned to a specific joint or joints, a process called *skinning*. This defines how the vertex is to move as the joint or joints are adjusted. Next, a level of movement systems is placed on specific joints and bones. These allow multiple bones and joints to be moved by a single control, rather then adjusting them individually. Finally, the highest level in the hierarchy is created, the user controls. These controls directly manipulate the movement systems through respective attributes on the control. When the animator selects a control, a list of its attributes is displayed. The animator may then adjust these attributes driving the underlying rig without ever dealing with the rig itself. This simplifies and eliminates many problems which may occur from handling the rig directly.

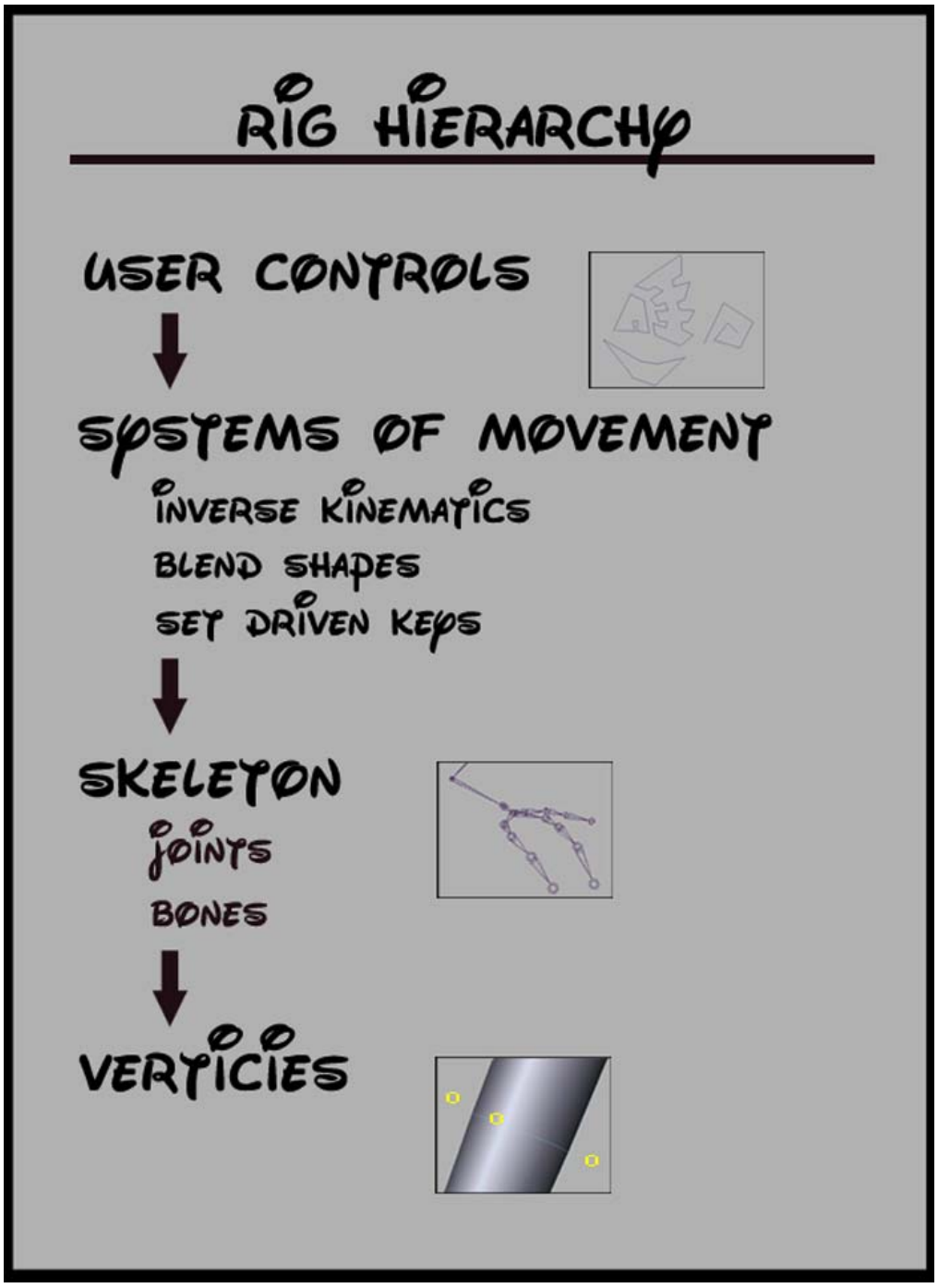


Fig. 4. Hierarchy of a rig.

III.4.1. The Body

The rig for Sparky's body was centered around one joint, appropriately titled the *root joint*. It was ultimately responsible for any movement involving the entire character. All other joints in the skeleton were parented to this root joint. To keep the rig consistent, and for ease of use, each joint was oriented to move in the manner described by Fig 5.

Rotation	Translation
x-axis = pitch or bend	x-axis = side-to-side
y-axis = yaw or side-to-side	y-axis = up and down
z-axis = roll or twist	z-axis = front-to-back

Fig. 5. Joint orientation in the rig's skeleton.

This allowed the animator to intuitively know how the geometry would react when moving any given joint. Fig. 6 shows the complete underlying body rig. It also depicts how the final geometry was related to it. All geometry, excluding tongue and hands, were parented to their respective joints. The tongue and hands, as discussed later, were skinned.

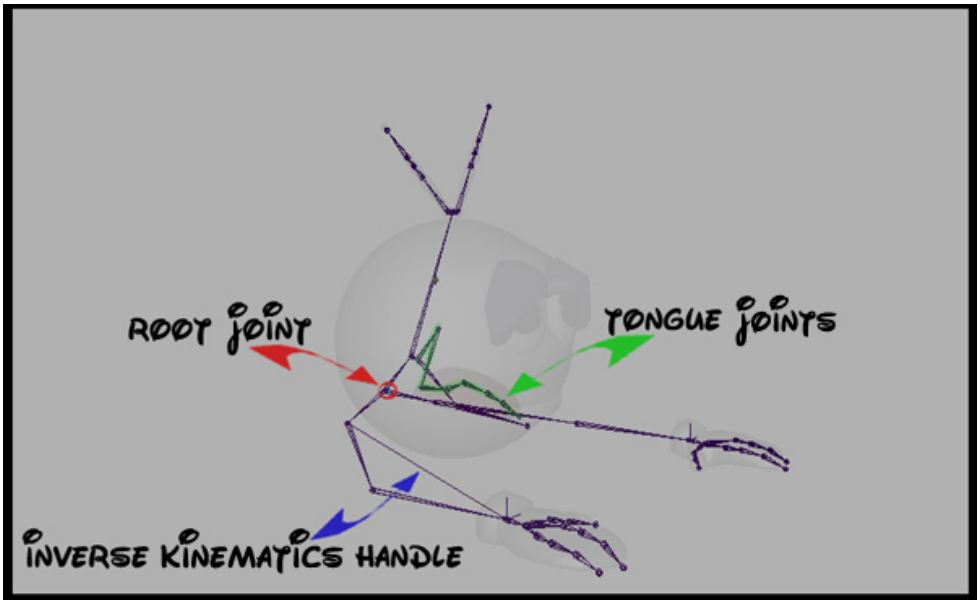


Fig. 6. Underlying rig detail.

Although the character had no visible arms, arms were built into the rig so that any hand movements would look physically accurate. This also gave the animator a frame of reference for realistic arm motions for the character. An inverse kinematics system [8] was set up on the arm joints to further ease animation. This meant that the animator could position the hand as desired without having to worry about moving the elbow and shoulder joints as well.

Finally the user controls labeled in Fig. 7 were built. A *control* is any object designed to act as the driving force on a particular part of the rig. Specific joints or joint

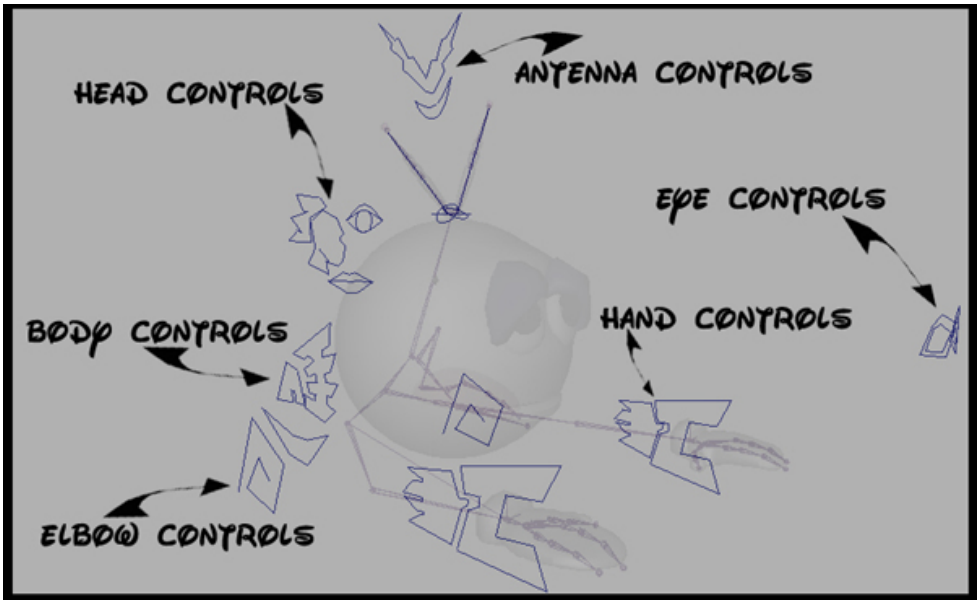


Fig. 7. User controls diagram.

actions are constrained to specific attributes of these controls. This enables the controls to drive the rig. Because splines [8] can be quickly calculated and easily shaped, they were chosen to comprise the control "handles" for Sparky. When one of these spline control "handles" was selected, a list of the attributes specific to that control was displayed. Fig. 8 illustrates a selected hand control and the attribute box displayed upon selection. The animator worked solely with these displayed attributes, adjusting, refining and setting their values. The various controls were selected and adjusted until the desired animation results were achieved.

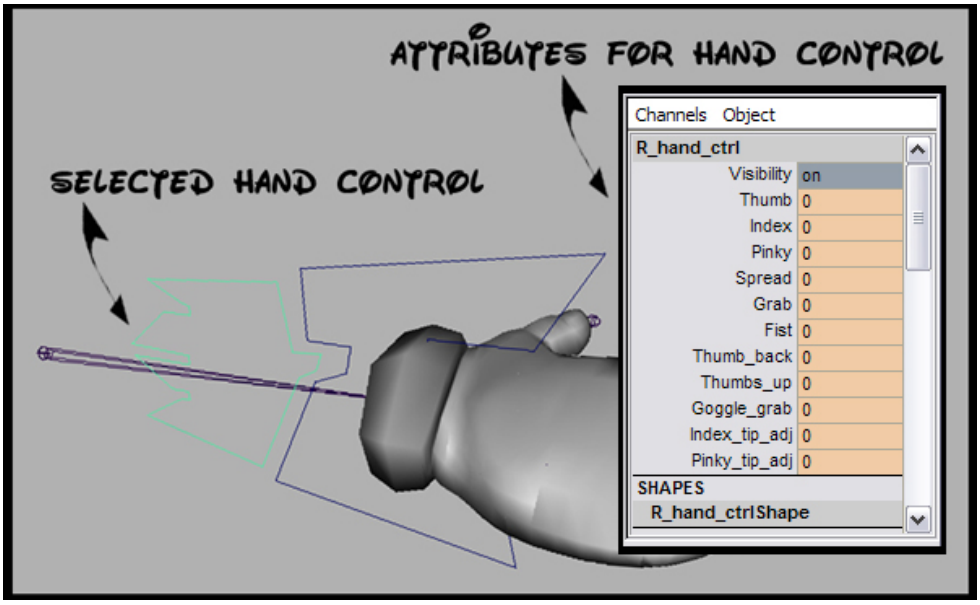


Fig. 8. Right hand control detail.

III.4.2. The Hands

The joints in the hand rigs were *skinned* to the geometry. Skinning is the process of attaching geometry to joints, allowing the geometry to deform based on the movements of the joints [2]. Each vertex of the geometry could then be adjusted until the movement looked correct. Two driving controls were created for the hands. A larger "arm_control", seen in Fig. 9, was for hand translation and wrist rotation while the smaller "hand_control" moved the fingers. The "arm_control" drove the hand translation through movement of an underlying inverse kinetics handle. All other hand movement was driven through a process called *set driven keys* [8].

Set driven keys allow a control attribute to drive a preset movement. By simply adjusting the attribute's value, these preset movements are blended together or interpolated. Fig. 9 demonstrates the power of set driven keys on the hand. Each position displayed is driven by only one attribute on the smaller "hand_control". Rather

then positioning the fifteen hand joints individually, the animator has only to adjust one control attribute.

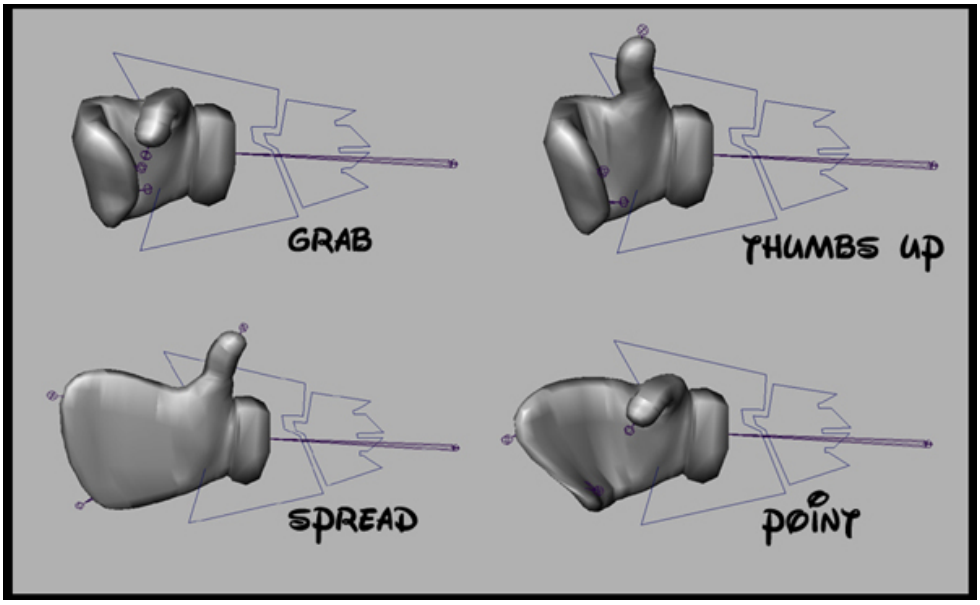


Fig. 9. A few sample "set driven key" hand positions.

III.4.3. The Eyes

The eyes were controlled a little differently. No joints were used in this part of the rig. Instead, the eyes were controlled through *expressions*, segments of run-time code that use mathematical functions to relate object attributes. The expressions utilized the fact that Sparky's head was spherical. Each pupil was built so that its rotation pivot lay at the exact center-point of the head. The pupil could then be rotated in any direction and always appear to "stick" to the surface of the face. Thus, expressions constraining the translation of the "eye_control" to the rotation of the pupils were created. Upper and lower bounds were enforced so that the pupils were limited to movements within the eye whites. This caused the eyes to move, following the movements of the "eye_control."

The pupils were also given the ability to move independently of one another, adding to the animator's control.

The eyelids were driven completely by *blend shapes* [8]. Blend shapes, described in more detail later, allow a piece of geometry to interpolate, or morph, between two or more previously defined shapes. Therefore, an original pair of eyelids was built. They were modeled in what was to be their "open" position. From duplicates of these open eyelid shapes, other positions like closed, wide-eyed or sad-eyed could be modeled. Finally, through the use of a control, blend shapes were set. The result allowed the eyelids to transition smoothly between any of the positions.

III.4.4. The Props

Occasionally, Sparky was to interact with props acquired from behind his back. When a CG character must grab or hold an object, special systems must be designed to create the proper illusions of touch and grasp. In computer graphics objects are made up of equations resulting in "virtual surfaces." Unless a system of collision detection is in place, there is no way for the computer to know whether or not two surfaces are touching. Therefore, it is up to the animator to create this illusion. A system to do just that was incorporated into the rig design. A "props" attribute was added to the root control. An expression allowed this attribute to flip through the five various props as needed. The expression constrained a locator attached to the prop, to a permanent locator positioned at Sparky's wrist. This resulted in the wrist motion driving the prop. Each time a new integer, corresponding to a particular prop, was entered, the new prop would appear

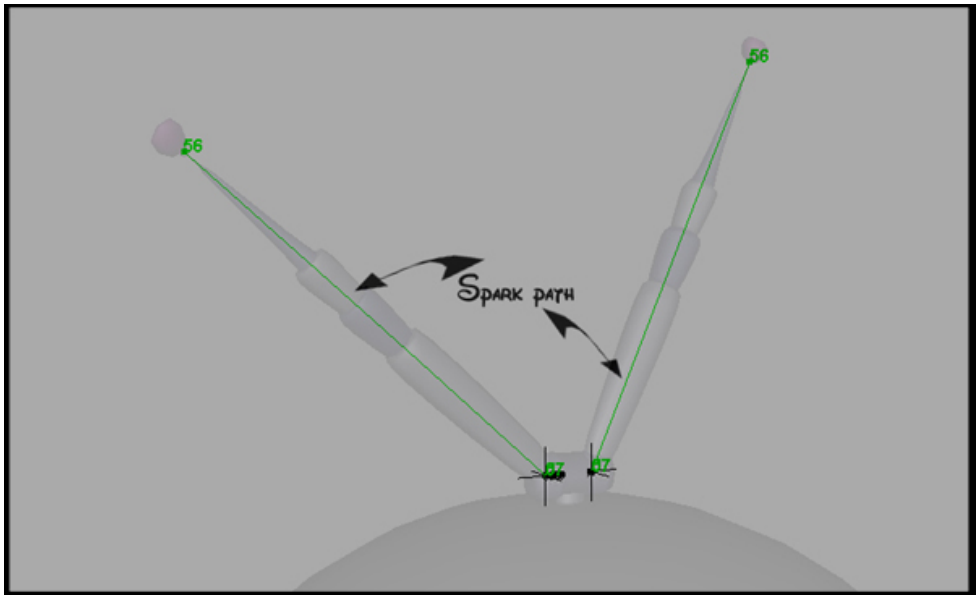
"attached" to Sparky's hand. In addition, the hand would simultaneously be updated to the appropriate looking grasp for that prop.

III.5. Rigging of Secondary Features

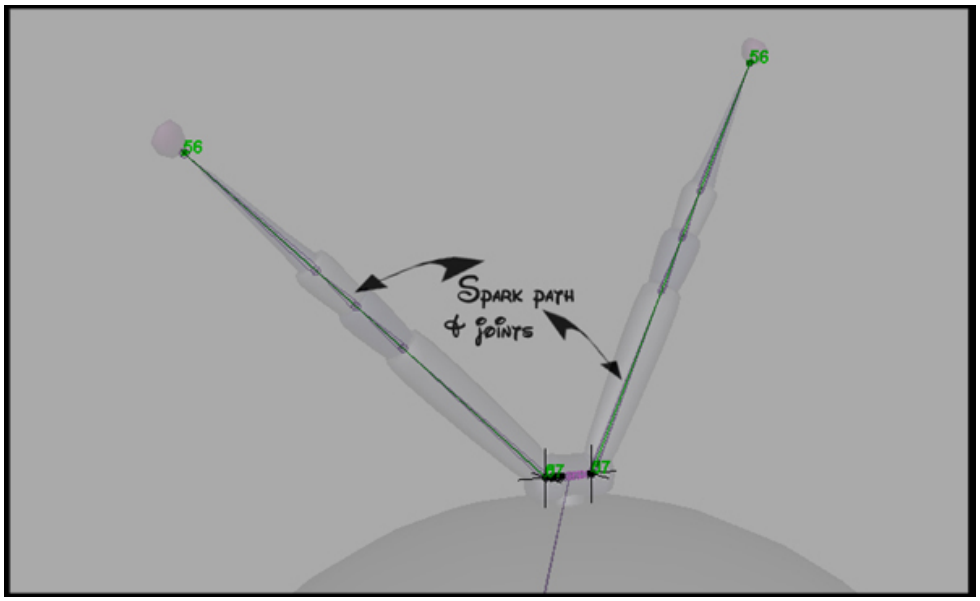
A major advantage of doing animation with computers is that much of the work, previously done by hand in traditional 2D animation, can now be set up for the computer to do. The more features that can be successfully automated by the computer the less time that must be spent by the animator later. While these automated features may take some time to initially set up, they will save much more time in later stages of the process. All of the secondary features in this character were fully automated, while leaving the animator full control over the timing of these features.

III.5.1. The Spark

The spark was created to enhance the character visually. It originates at Sparky's antenna base, running upward along the antenna to dissipate upon reaching the antenna tips. The tips, in turn, would then light up. The spark was animated completely by expressions. This was done in an effort to ease the animator's workload. Because the antenna could be in any number of positions in 3D space, the spark had to be constrained so that each end would follow its respective path up the antenna to the antenna tip no matter what position the antenna assumed. To achieve this, a set of spline motion paths, shown in green in Fig. 10(a), were created. They ran from antenna base to tip. Each path was constrained to its respective set of antenna joints as seen in Fig. 10(b). This caused



(a) Spark end's motion paths.



(b) Antenna joints and path.

Fig. 10. Antenna rig details.

the motion path to deform and readjust simultaneously with the antenna, as the antenna joints were moved through space.

A locator attached to each end joint of the spark, depicted by the red arrows in Fig. 11, was then constrained to the adjacent motion path, labeled by the green arrows. This fixed the spark to the antenna. However, as the spark rose toward the antenna tip, it needed the ability to shrink and expand in order to correctly span the changing gap. As Fig. 11 illustrates, the spark was an actual piece of deforming geometry. Therefore, each set of vertices on the spark could be moved independently to achieve the desired effect. An unparented joint, shown by the blue arrows in Fig. 11, was placed at the center of each group of twelve vertices. The vertices, depicted in alternating groups of yellow and pink, were then attached to their center joint respectively. This pairing allowed for

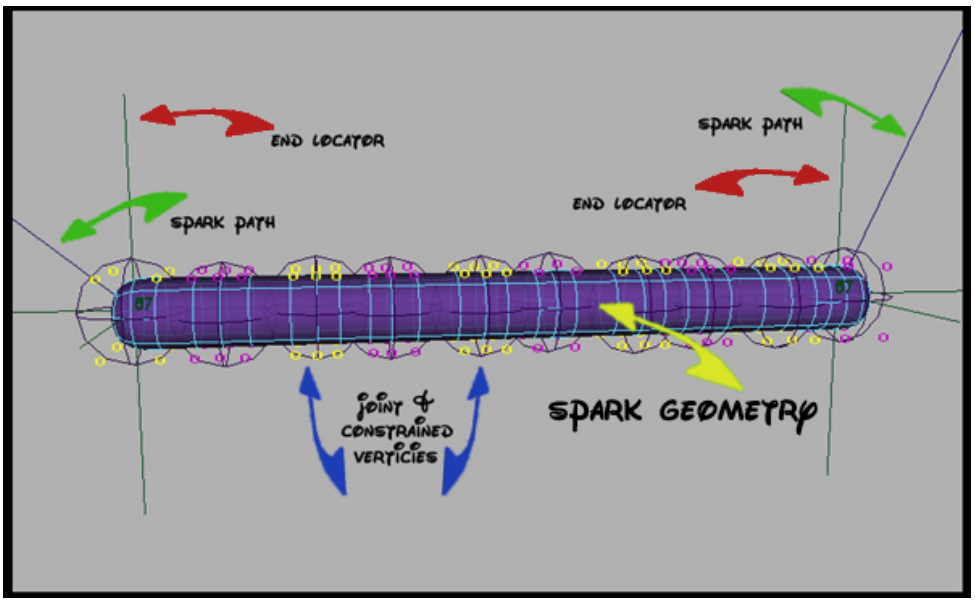


Fig. 11. Spark rigging detail.

smooth reshaping and deformation of the spark geometry. Next, an expression for frame-by-frame reshaping was created. Fig. 12 shows the expression code for calculating the

frame-to-frame change in the y-axis. First, the current locations of the spark's two endpoint locators along their respective motion path were determined. Then the space between the motion paths was calculated and stored. Each joint was finally adjusted according to its position along the antenna. This resulted in a spark that smoothly spanned the space between the antennas in the current frame. Note that although the code is not displayed here, this was also done for the x and z-axes.

```
//**sample of spark adjustment expression in the Y-axis
//location of the spark endpoint along the motion path
  float $LY = L_tenna_motion_path_lctr.translateY;
  float $RY = R_tenna_motion_path_lctr.translateY;

//distance between left and right antenna at present frame
float $newPosY = $LY - $RY;

//y-axis spacing adjustment of each joint
 if ($newPosY != 0)
   {
     $YposAdj = ($newPosY/4);
     float $TY4 = (-2 * $YposAdj);
     setAttr "spark_joint_4_TY_grp.translateY" $TY4;
     float $TY3 = (-1.5 * $YposAdj);
     setAttr "spark_joint_3_TY_grp.translateY" $TY3;
     float $TY2 = (-1 * $YposAdj);
     setAttr "spark_joint_2_TY_grp.translateY" $TY2;
     float $TY1 = (-0.5 * $YposAdj);
     setAttr "spark_joint_1_TY_grp.translateY" $TY1;

     float $TY5 = (2 * $YposAdj);
     setAttr "spark_joint_5_TY_grp.translateY" $TY5;
     float $TY6 = (1.5 * $YposAdj);
     setAttr "spark_joint_6_TY_grp.translateY" $TY6;
     float $TY7 = (1 * $YposAdj);
     setAttr "spark_joint_7_TY_grp.translateY" $TY7;
     float $TY8 = (0.5 * $YposAdj);
     setAttr "spark_joint_8_TY_grp.translateY" $TY8;
   }
```

Fig. 12. Excerpt of y-axis MEL code from the spark expression.

The final step taken was to give each joint in the spark an alternating, random sine function. This was done to recreate the arbitrary nature in which a spark's shape is continuously changing. Fig. 13 depicts the expression used. The joints were given a sine function driven by the current frame number. The period of the function, or cycle time

was an experimentally value. Each joint's sine function also included a random number selected from within a predetermined range. The random number accounted for the vertical phase or height of the sine function cycle. Every time the joint completed one period, or cycle in the sine wave, a new random number would be substituted. This was

```
//**joint 4 sine wave function

//gets stored random number for the cycle
  float $randNum = spark_joint_4_sin_grp.rand_num;

//back to the start of a new cycle…so get new random number
  if ((spark_joint_4_sin_grp.translateY > -0.015) &&
(spark_joint_4_sin_grp.translateY < 0.015))
    {
      $randNum = rand(.03,.19);
      setAttr "spark_joint_4_sin_grp.rand_num" $randNum;
    }

//sine function for joint4
   spark_joint_4_sin_grp.translateY = spark_joint_4_sin_grp.rand_num *
sin((.35*(frame+1)));
```

Fig. 13. Excerpt MEL code from joint 4 of the spark's sine function expression.

done to create a unique looking spark each time. In addition, another sine function was added to set mirrored pairs of spark joints to create an even more natural look.

Once the spark reached its destination, the antenna tips, the spark automatically reset itself and an expression to "light up" the antenna tips took over. Attributes of the antenna tips shader were linked through Maya's connection editor [8] to attributes created in the *antenna control*. These attributes were in turn controlled through the expression shown in Fig. 14, resulting in the appearance of the antenna tips "lighting up." The expression began by collecting and storing the frames that would eventually comprise the entire "light up" cycle. A start, mid and end point frame completed this cycle. The mid

```
 //**expression creating the antenna tip glow (only start points)
//get start mid and end points of the "light up" cycle
    int $LitFrameNumBegin = (frame + antenna_ctrl.spark_end_frame);
    int $LitFrameNumMid = ($LitFrameNumBegin + 9);
    int $LitFrameNumEnd = ($LitFrameNumMid + 10);

//set up the light keyframes and adjust the animation curves for them
//start points (for each attribute)
    currentTime ($LitFrameNumBegin);
    setAttr "antenna_ctrl.r_light_intensity" 0;
    setKeyframe "antenna_ctrl.r_light_intensity";

    setAttr "antenna_ctrl.l_light_intensity" 0;
    setKeyframe "antenna_ctrl.l_light_intensity";

    setAttr "antenna_ctrl.tip_transparency" 0;
    setKeyframe "antenna_ctrl.tip_transparency";

    setAttr "antenna_ctrl.tip_glow" 0;
    setKeyframe "antenna_ctrl.tip_glow";

    setAttr "antenna_ctrl.tip_translucence" 0;
    setKeyframe "antenna_ctrl.tip_translucence";

//adjust the curve tangents in the animation graph for each attribute
//start points (flattening the curve at the start point)
selectKey -clear;
selectKey -add -k -t $LitFrameNumBegin antenna_ctrl_r_light_intensity;
keyTangent -lock off;
selectKey -clear;
selectKey -add -ot -t $LitFrameNumBegin antenna_ctrl_r_light_intensity;
keyTangent -itt flat -ott flat;

selectKey -clear;
selectKey -add -k -t $LitFrameNumBegin antenna_ctrl_l_light_intensity;
keyTangent -lock off;
selectKey -clear;
selectKey -add -ot -t $LitFrameNumBegin antenna_ctrl_l_light_intensity;
keyTangent -itt flat -ott flat;

selectKey -clear;
selectKey -add -k -t $LitFrameNumBegin antenna_ctrl_tip_transparency;
keyTangent -lock off;
selectKey -clear;
selectKey -add -ot -t $LitFrameNumBegin antenna_ctrl_tip_transparency;
keyTangent -itt flat -ott flat;

selectKey -clear;
selectKey -add -k -t $LitFrameNumBegin antenna_ctrl_tip_glow;
keyTangent -lock off;
selectKey -clear;
selectKey -add -ot -t $LitFrameNumBegin antenna_ctrl_tip_glow;
keyTangent -itt flat -ott flat;

selectKey -clear ;
selectKey -add -k -t $LitFrameNumBegin antenna_ctrl_tip_translucence;
keyTangent -lock off;
selectKey -clear ;
selectKey -add -ot -t $LitFrameNumBegin antenna_ctrl_tip_translucence;
keyTangent -itt flat -ott flat;
```

Fig. 14. Excerpt of antenna tip glow MEL expression.

and end point frames were a predetermined number of frames after their respective starting frame. Next, key frames were set for all the attributes that were to animate during the cycle. Once set, the animation curves for each of these keyed attributes were adjusted. This created a smooth, more believable looking "light up" cycle. Note that Fig. 14 only provides code for the attributes of the starting key frame of the cycle. The same calculations were also done for both the mid and end point frames.

Finally, the entirety of the spark cycle was abstracted into two simple attributes, as seen in Fig. 15. The first attribute called "Spark_end_frame," enabled the animator to adjust the number of frames it took for the spark to complete its cycle. For the

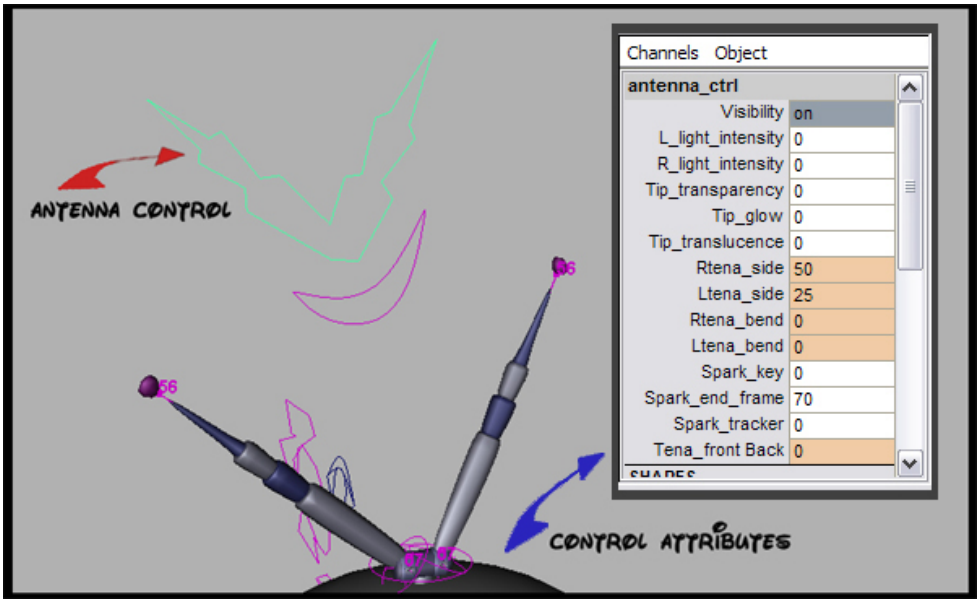


Fig. 15. Antenna control and attribute list detail.

animator's convenience, the cycle was set to a predetermined default value of seventy frames. The second attribute, "Spark_key," acted as a simple toggle switch indicating when the expression setting up a cycle was to be run. To operate, the animator had only

to move to the frame she wanted the spark cycle to begin, adjust the “Sparke_end_frame” attribute if desired, and enter the value 1 into the “Spark_key” attribute.

III.5.2. The Plasma Shell

The “plasma” shell around Sparky’s body was also created as a fully automated, visual enhancement to the character. It consisted of one procedurally-shaded polygonal sphere, point constrained to the body root joint. To create plasma-like movement, four animatable deformers were attached. The first group, denoted by blue arrows in Fig. 16, was comprised of three *wave deformers*. A wave deformer is an animation tool provided in the software package Maya [8]. Each deformer operates along a single axis, changing the location of attached vertices according to a set sine wave function. One deformer was placed along each axis of the group where they would be animated later. Their collective movements gave the plasma sphere an off-balance, jello-esque appearance. The best orientation for these deformers was determined experimentally.

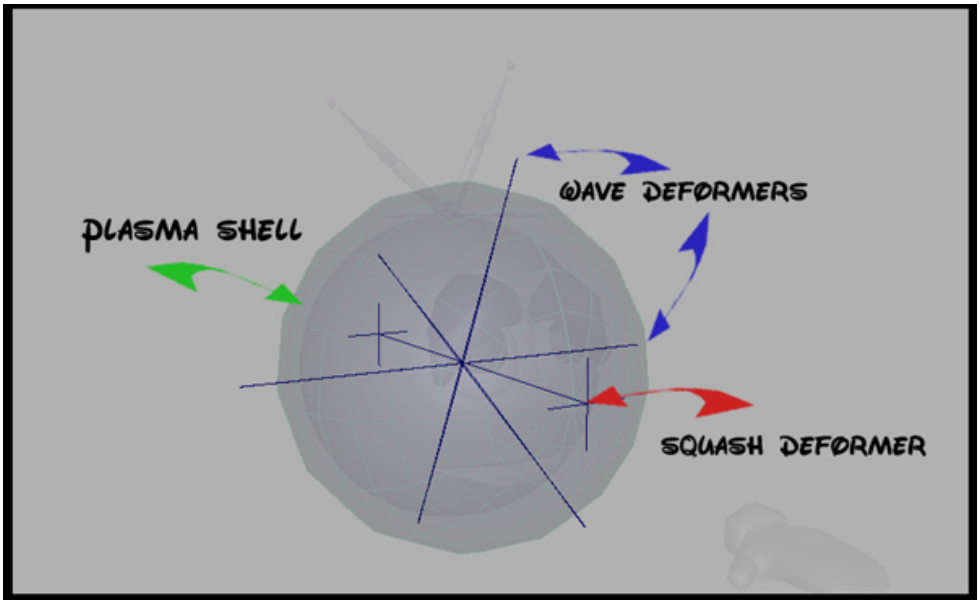


Fig. 16. Plasma shell deformation rig.

They were animated in a test loop and found to produce the best visual effect when set at a thirty degree angle about the camera's z-axis. A fourth and final squash deformer was then attached. This created a little more variation in the plasma sphere's look as well as adding a subtle growing and shrinking effect.

Next, an expression initializing and controlling the sphere's deformer animation was created. Fig. 17 details this expression for the first deformer, *wave handle 1*. As seen below, the expression began by returning to frame zero and removing any previously set animation. It then iterated a number of times specified by the animator.

```
//**deletes the old and sets the new animation loop for the plasma
//**shell. This shows expression ONLY for wave handle 1.
  //initializations and move to frame 0
     int $n = 0;
     int $frameNumber = 0;
     int $shellIt = 0;
     $shellIt = root_ctrl.shell_iterations;
     currentTime 0;

  //deletes any old animation that may have been set previously
     CBdeleteConnection "wave1.off";

  //loop setting up each cycle for specified number of iterations
     for ($n=0; $n<($shellIt); $n++)
       {
  //moves to frame
        currentTime $frameNumber;

  //clears key frame if exists and sets the new one for wavehandle 1
        select -cl  ;
        select -r wave1Handle ;
        select -addFirst wave1 ;
        timeSliderClearKey;
        setAttr "wave1.off" 12.4;
        setKeyframe "wave1.off";

  //updates frameNumber and moves to that frame
        $frameNumber = $frameNumber + 99;
        currentTime $frameNumber;

  //clears key frame if exists and sets the new one for wavehandle 1
        select -cl  ;
        select -r wave1Handle ;
        select -addFirst wave1 ;
        timeSliderClearKey;
        setAttr "wave1.off" 0;
        setKeyframe "wave1.off";

  //update frame number for cycle of 100 frames
        $frameNumber = $frameNumber + 1;
      }
  //reset the expression switch
        setAttr "root_ctrl.key_shell" 0;
   }
```

Fig. 17. Excerpt from plasma shell animation MEL expression.

Each iteration set up one animation cycle lasting about 200 frames. While in the loop, key frames were set for the various sphere deformers at predetermined intervals. The animator could control all of this through a few attributes added to Sparky's *root control*. The attributes and control can be seen detailed in Fig. 18. Knowing that an iteration was

approximately 200 frames, the animator could adjust the “Shell_iterations” (“$shellIt” in the code of Fig. 15) attribute as desired. If the shell already had an animation cycle set,

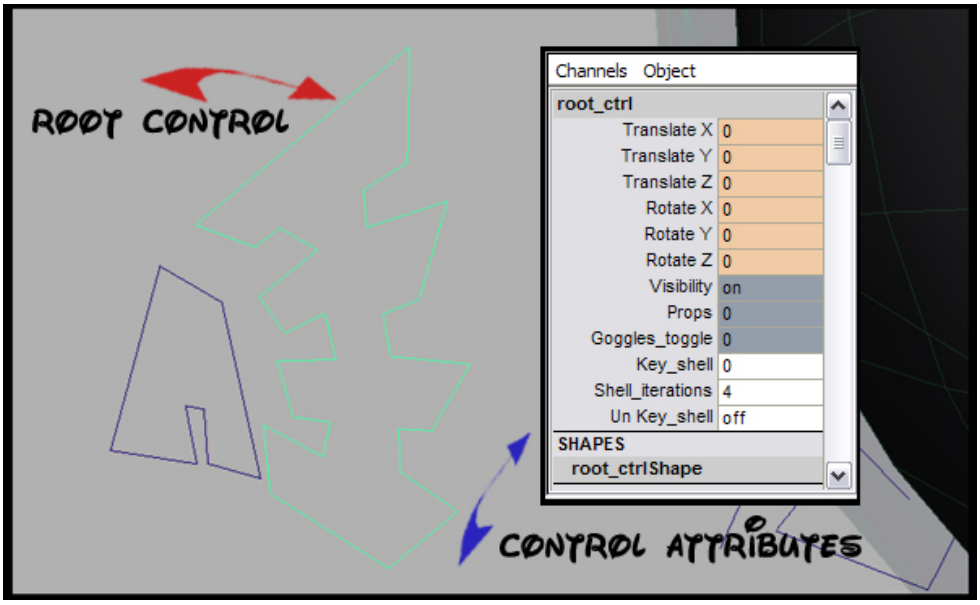


Fig. 18. Root control and plasma shell attributes detail.

the animator could turn “UnKey_shell” on to delete it before setting a new cycle. Finally, all the animator had left to do was type the number “1” into the “Key_shell” attribute and a new cycle would set itself.

III.5.3. The Float

The float was an automated secondary feature used to keep Sparky subtly “alive” and in character. In animation, when a character completely stops moving on screen it is immediately noticeable. Therefore, characters must always be kept moving, even if the motion is quite subtle. The automated float ensured that Sparky would never go “dead” on screen. He would, at some level, always be moving. This is consistent with his character. He was being viewed at the molecular level, so he should be floating in space.

The float expressions, seen in Fig. 19, once again used a simple sine function driven by the frame number. Visually predetermined values were set for the period and phase of the functions. The expressions were then attached to Sparky's head and hands

```
//**expressions used to create the float

//Left arm sine float expression
  L_arm_float_grp.translateY = .23 * sin((.07*(frame-3)));

//Right arm sine float expression
  R_arm_float_grp.translateY = .23 * sin((.07*(frame-3)));

//body sine float expression
  body_float_grp.translateY = .16 * sin((.07*frame));

//head bob sine float expression
//this passed into an attribute connected to the character's nod attribute
  body_float_grp.float_head_bob = -3 * (.16 * sin((.07*(frame-3))));
```

Fig. 19. Automated float MEL expressions.

independently. This allowed them to be offset by three frames, producing a more acceptable motion. In addition, a very subtle nod expression was added to the head. It was offset, based on the cycle of the sine function, to again create more acceptable looking movement.

III.5.4. The Toggle Switch

The toggle switch was introduced to save on real-time interaction within the software program being used. In the process of CG animation, animators manually manipulate the character using an appropriate software interface [8]. Frame by frame they create the desired actions for each movable aspect of that character. However, expressions built into the rig are often recalculated and updated when the animator changes a frame. The time taken to do this will slow interaction dramatically, limiting

the amount of work the animator can accomplish. Thus, a toggle switch was devised to disable extraneous calculations. When the animator wanted to experience the quickest interaction times, she could simply flip the switch attribute on. This would, in effect, shut off all secondary feature expressions from being calculated and updated in real-time. Conditional statements testing toggle switches were added to the expressions of the secondary features. These, in turn, could be controlled by simple attributes, acting like a switch, added to Sparky's *all control*. The initial attribute created was a master switch. It toggled all secondary feature expressions on and off. An additional switch was later implemented, disabling the float feature. This allowed for the float feature to be independently adjusted, thus creating a greater range of control for the animator.

III.5.5. The Lip-Synch

In her recent thesis, Haaser [5] described a system for real-time automation of lip-synch for animation. The thirteen phoneme shapes she depicted were used as references for the mouth shapes created for Sparky. Each shape was individually modeled from a duplicate of the original head. Then blend shapes between the original head and the phoneme heads were created. Blend shapes work much like set driven keys. Upon creation, the animator simply adjusts a resulting attribute to blend between the original face and the desired phoneme shape. Multiple phoneme shapes may also be blended together to achieve any number of varying faces. These blend shapes were then set up to be driven by a special phoneme control. The attributes of this control not only drove the phoneme shape, but it moved the tongue and teeth as well. When set to its highest value, typically "1," each attribute moved Sparky's mouth, teeth and tongue, to a specific

phoneme shape. This resulted in one of the thirteen possible phoneme shapes depicted in Fig. 20. The animator had only to key the appropriate phoneme shape at the appropriate frame in the animation, according to the dialogue. Initially, an automated system using Haaser's code was attempted to eliminate this step in the process. However, due to time constraints and code limitations the job eventually had to be manually completed by the animator.

As stated previously, the teeth and tongue were driven by the phoneme control. The teeth were parented to a group of jaw joints in the rig. Like a human skull the upper teeth never moved. However, the lower teeth moved about a pivot point set by the jaw joint in the back of the mouth. The tongue was rigged much like Sparky's hands. It was skinned to a set of tongue joints parented to the main jaw joint. Skinning allowed the tongue to deform into all the shapes necessary for speaking. The teeth and tongue were then manually adjusted and set appropriately for each phoneme shape.

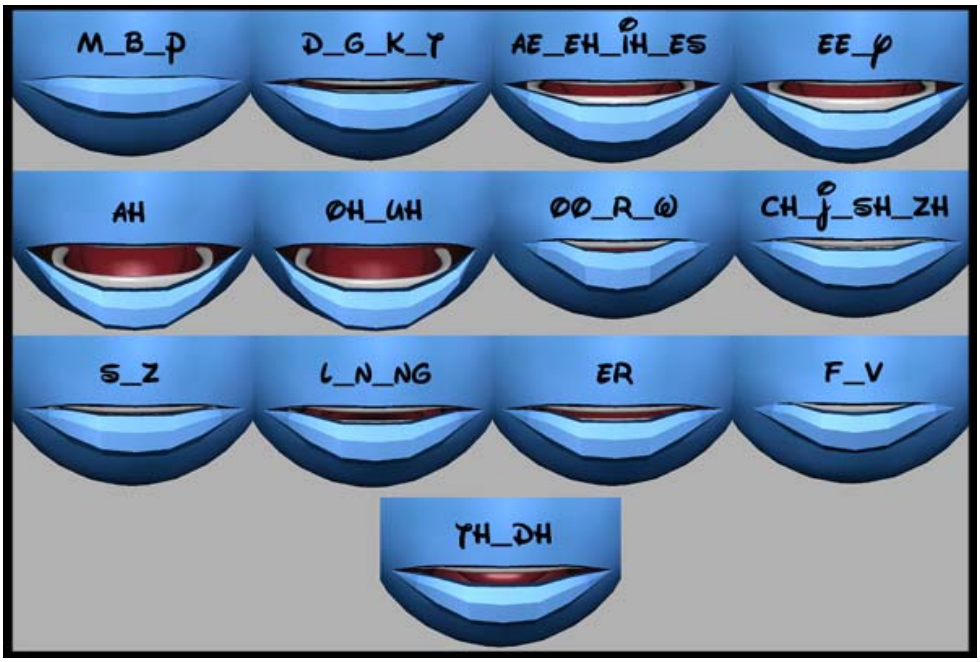


Fig. 20. Phoneme shape details.

III.6. Lighting

There were two levels of lighting created for the scene. The first, comprised of twenty-six low intensity lights, provided the ambient lighting. These lights were arranged in a large spherical configuration pointing inward. Their purpose was to subtly define each objects shape, most noticeably enhancing edges and wrinkles in the geometry. The second level of lights provided the key and fill lighting [8] for the character. Objects like the plasma layer and eye whites required unique levels of illumination. They needed lights designed and linked specifically to them, while other similarly shaded objects shared lights. A few specialty lights were also vital to the scene. Soft blue lights placed at the center of Sparky's body were aim-constrained to the hands and antenna. These simulated the glow light originating from Sparky's body. A set of lights was also constrained to the antenna tips. These lights, animated in tandem with the antenna tips' glow, cast light on the rest of Sparky's body. Finally, special lights, simulating the specular highlights for each eye were created. These were constrained to the eyeballs, enabling the eyes to have clearly defined specular highlights in any position.

III.7. Shading

There were a number of shaders created for the scene. Because of the cartoon-like style, most were limited to simple shaders with procedural textures and bump maps. Procedural textures and bump maps were easy to set up in Maya and computationally quicker then image based maps. Ray tracing was excluded to keep render times low. Instead, reflections were done exclusively with environment maps. Motion blur was also avoided after several render tests produced mediocre results and lengthened render times.

Special UV mapping was also avoided whenever possible. Only Sparky's head and goggles eventually required anything other than Maya's default UV mapping. A 2D post-process glow was added to several scene objects including Sparky's body. These layered shaders, with their automated post-process glow, were the most complex shaders used.

III.8. Animation

Upon completion of the dialogue track, there was just over five minutes of animation to be done. To begin, rough *animatics*, a set of quickly depicted character poses timed to the vocal track, were made. These were done using a *pose-to-pose* approach, which set up key character positions timed to the dialogue. This helped to sort out the timing and animation flow throughout the story. The animatics, in conjunction with more detailed character pose sheets like Fig. 21, created the resulting sequence of action. A *straight ahead* approach was taken with the majority of the animation. The character was animated from one complete pose to the next rather then setting up all of the rough poses and refining them. In this case, the straight ahead approach allowed for more freedom in adjusting poses. It also allowed for on screen pose locations to be redefined as needed. Posing the character began with the arm movements. Once set, the head movement was put in place followed by the entire body movement. Each scene was animated at this level first. Next, a *layered approach* was used. The eyes and eyelids were animated. Subsequently, the lip synch was finalized and all of the secondary features were turned on and set appropriately.

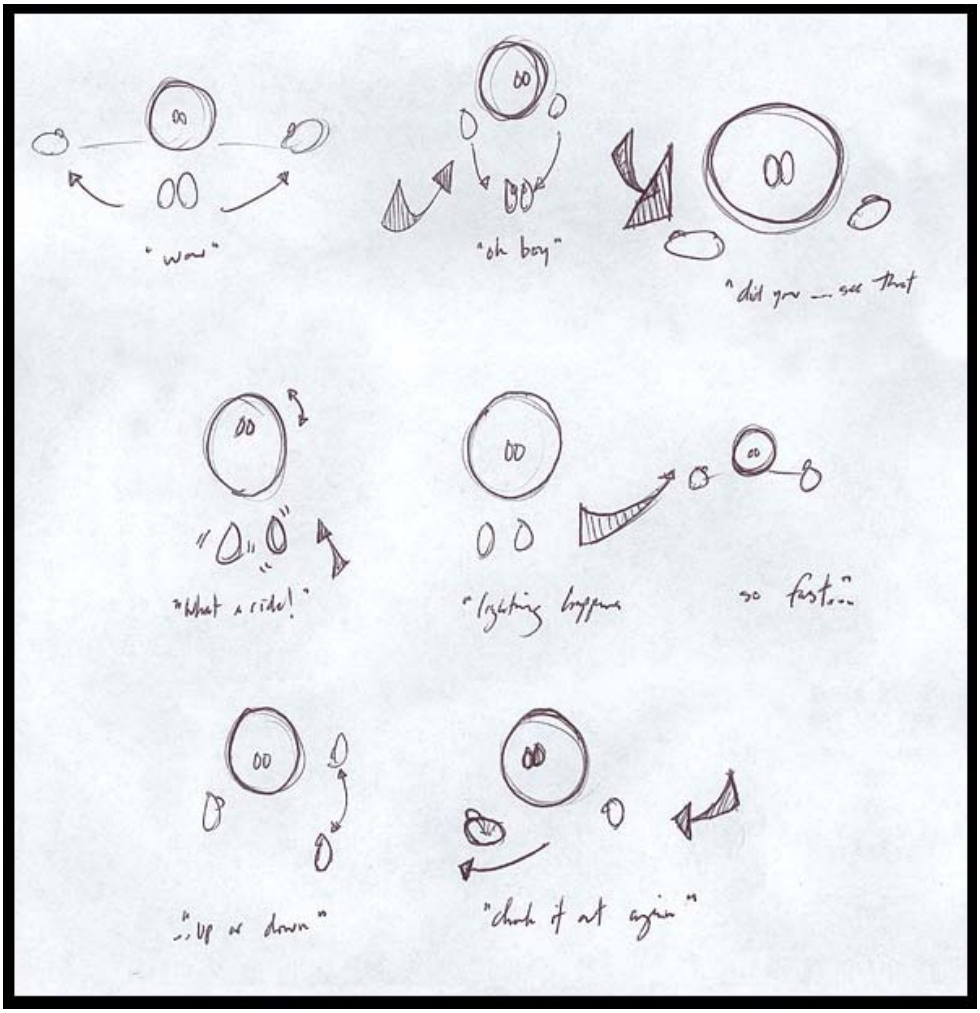


Fig. 21. Character pose sheet.

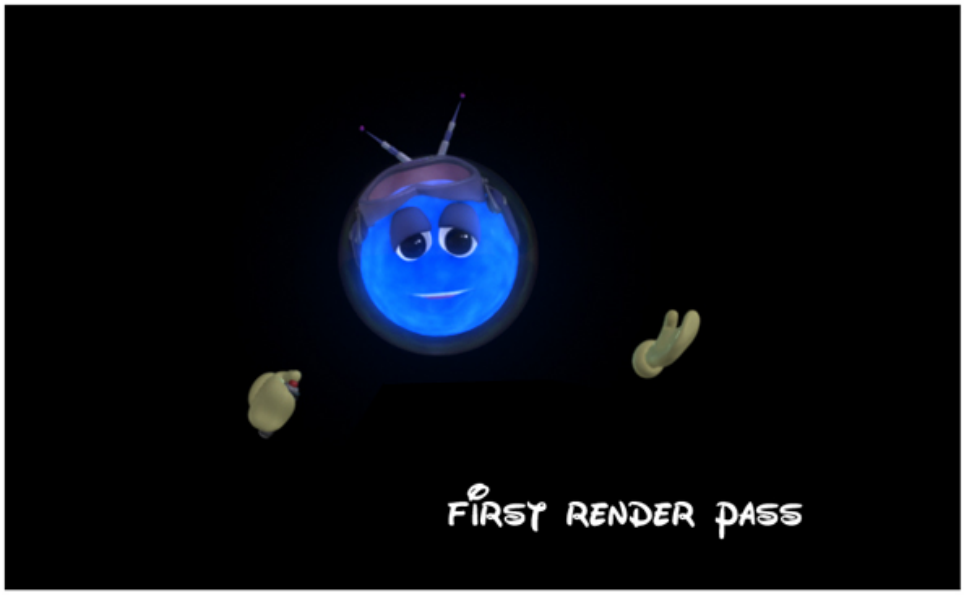
III.8.1. The Trax Editor

The Trax Editor is an animation editor provided in Maya [8] that allows the animator to reuse animation segments, called *clips*. It provides a way of copying and recycling segments of previously completed animation. This editor was used throughout the animation process. Clips were carefully selected from previous segments, and reused or blended with current animation. Because clips are combined with the current character's movement, the viewer doesn't realize he or she is seeing them again and again. Repeated motions, like Sparky straightening his antenna, were done using clips from the Trax editor. Some of the body motion was also reused. The character's expressions and gestures changed while the body repeated previously created sequences.

III.9. Rendering

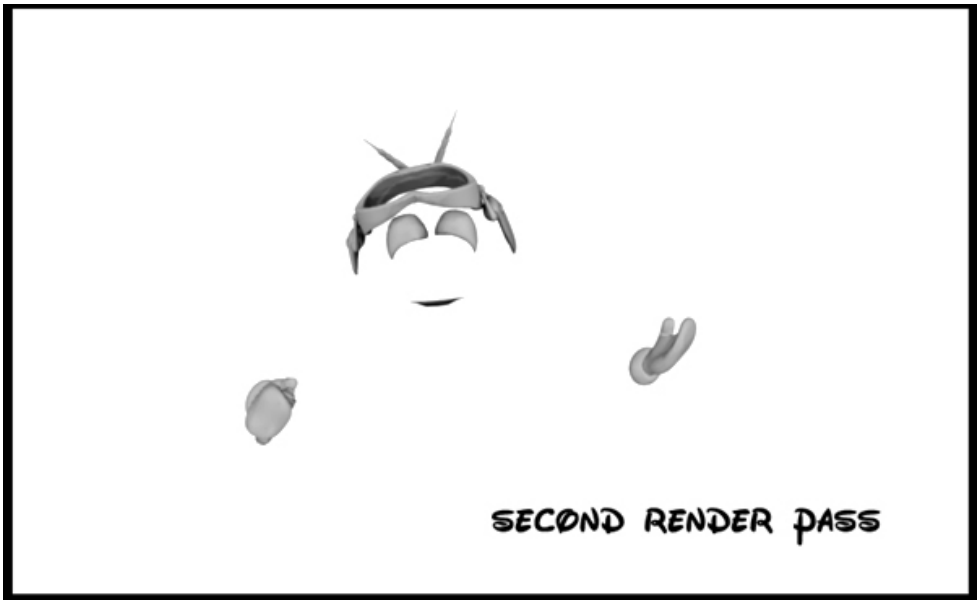
Three rendering passes were used. The first was the main lighting pass. It used all eighty-four lights and shaders as originally set up in the scene. The resulting render was a 720 x 486 color image like Fig. 22(a). Two additional passes were done to enhance the first one. The second pass was a faked *ambient occlusion* pass. Ambient occlusion is a term defining the blocked diffuse reflection of light, or the light blocked by other matte surfaces in the scene. Today's rendering software does not take this into account unless specified. This causes most images to look darker and most objects to look flatter then they should. However, true ambient occlusion calculations greatly increase final render times. Therefore, a second pass was done using the twenty-six low intensity lights previously discussed. With only these lights and a specially designed shader, a grayscale detail enhancing image was created. An example can be seen in Fig.

22(b). The third and final pass, Fig. 22(c), was strictly to aid in effective communication. Due to the glow applied to Sparky's body, his lower lip lacked definition at times. This pass rendered only a shadow for the lower lip, giving it better definition. It also aided in making the lip synch visually more effective. Finally, the second and third passes were composited with the original pass. This created the frames like Fig. 22(d) for the final composite image sequence. In all, more then 36,000 frames were rendered, stored, and composited.

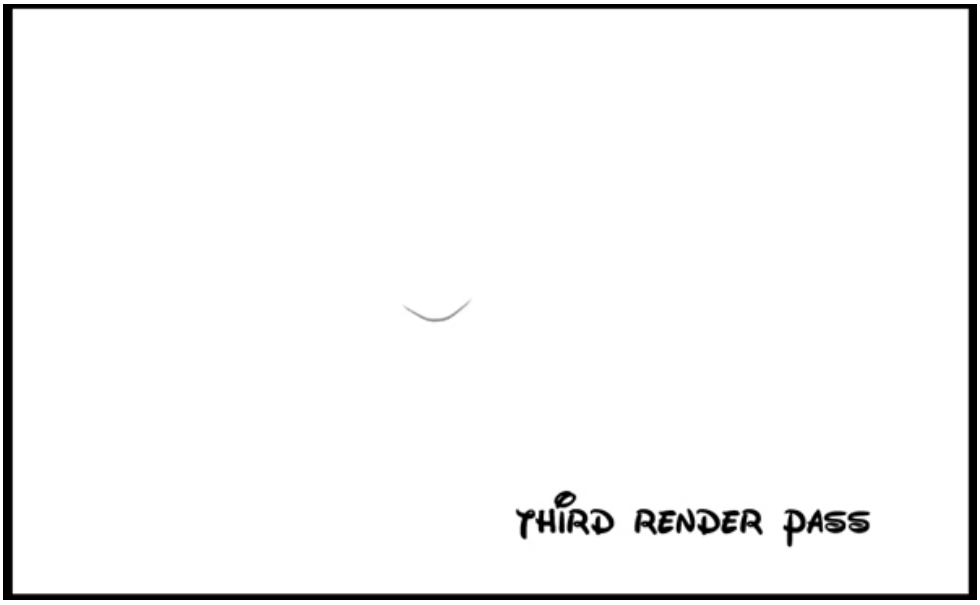


(a) Shaded color pass.

Fig. 22. Final rendered images.

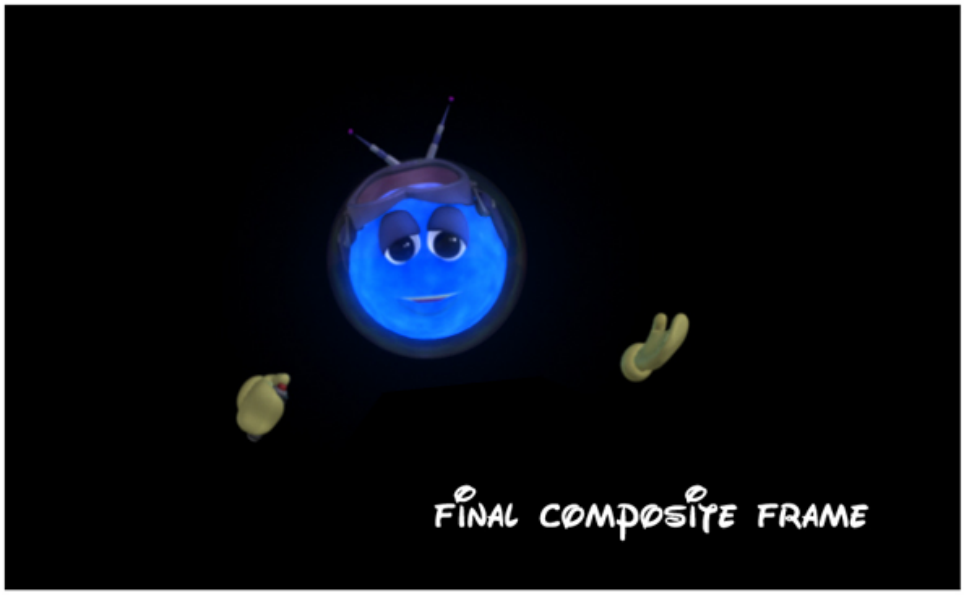


(b) Faked ambient occlusion pass.



(c) Lip shadow pass.

Fig. 22. Continued.



(d) Final composite.

Fig. 22. Continued.

III.10. Planetarium Specific

Because Sparky was part of a planetarium show, production had to consider and deal with a few unusual dynamics. In the dome, there are many opportunities for interactivity, both with the audience and with the show itself. Sparky was staged so that at times he could interact with either. This meant that all other media, such as slides, all-skies, panoramas and DIGI STAR II sequences, had to be preplanned so that Sparky could be animated accordingly. The animator had to know at all times what was going on in the dome around Sparky, so that Sparky's reactions and movements were appropriate.

To suit the planetarium context, Sparky was cast against a black background. The benefits of having "no" background were two fold. It would keep projected light to a minimum allowing other hardware, like the DIGI STAR II or slides, to be used

simultaneously with the character. It also lent itself to audience believability. The illusion of the character not being "framed" in any set area, allowed for a more believable use of interactivity. The audience would feel that Sparky was free to go anywhere he wanted in the dome, even though he was actually constrained to the fixed video screen. This meant he could interact much more naturally with other media displayed and with the viewing audience.

CHAPTER IV

EVALUATION

The evaluation of the methods used in this thesis project was informal. Each process described in the Methodology section was judged, based on personal experience and/or by records kept during production. Techniques implemented throughout were contrasted, comparing the potential time saved to the actual time saved. The trade-off of aesthetic quality to production efficiency was also considered, attempting to keep both as high as possible. Any aspects, which are standard time-saving techniques or made no significant alteration to production times, were not discussed.

IV.1. Conception and Design

Many of the choices made during the conception and design phase saved time throughout the later stages of production. These choices were the creation only one character, building only the parts of the character truly needed to convey the story, and limiting the background. All of these initial design choices saved countless hours in all of the following production phases, but especially in the animation phase.

In an informal interview, Pixar animator Alex Orelle, explained that the animation requirement per animator for this industry powerhouse is four feet, or about three seconds per week [12]. The requirement is also character dependent. In other words, if there are two characters in a four foot scene, then the scene should be completed in two weeks.

This planetarium show called for a total of five minutes and ten seconds of animation. Working at the industry standard of three seconds a week, it would take a

single animator just over 103 weeks, or two years, to complete the animation! However, we had only eighteen months and one animator to complete the entire show. Further, the animator was also the designer, modeler, rigger, shader, lighter, and renderer. Therefore, good decisions like creating only one character and designing with simplicity had to be made immediately.

IV.2. Modeling

The time saved in the modeling process stemmed fully from the previous conception and design phase. Simple shapes and low resolution polygons were used as an effort to make real-time interaction, during the later animation phase, as quick as possible. A key time-saving device in this phase was the fact that the designs were kept simple. Very few objects needed to be modeled in detail or remodeled. Because simple shapes were combined to create more complex looking objects, models could be created quite quickly. This also allowed UV texture mapping to be avoided in most cases. Shaders could also be simplified due to the fact that there was minimal complex geometry. Also, the fact that the number of objects needed was kept to a minimum aided in speeding up the time it took to complete the modeling.

IV.3. Rigging

A number of steps were taken in the rigging phase to create time-saving systems within the character's rig. These systems would be utilized in the animation phase of the production. Some systems worked flawlessly while others actually wasted production time. A review of both is detailed below.

IV.3.1. Automated Secondary Features

The automated secondary features, consisting of the spark, plasma shell, and float, worked perfectly. While it took a few months to initially implement these systems, once finished the total time spent animating any of these features was negligible. Only about one hour was needed to control these features for the entire piece. This was in comparison to the hundreds of hours spent on primary animation, and to the fact that animating these features by hand would have proved extremely difficult. The toggle switches designed for the features were an added bonus. They did their job of speeding up interaction times. In fact, the toggle switches made interaction so fast that no time was wasted.

IV.3.2. The Lip-Synch

There were two things done to the lip-synch rig which actually saved animation time. One was the control built specifically for the lip-synch phonemes. It put all the phonemes together in one tidy location. This made animating the lip-synch fast and easy because the interface was compact and quick to operate. The second time-saving factor was the way each phoneme attribute on the control was rigged to drive the mouth, teeth and tongue. This reduced the number of controls needed to animate the lip-synch by two thirds.

However, even with these time-saving features, the biggest loss of time occurred during the rigging stage of the lip-synch. The loss came when attempting to create a fully automated lip-synch system using Haaser's [5] method. Her approach was designed for

real-time automation of lip-synch. Her system works by breaking a vocal track into tiny increments of time, specifically two sound samples per animated frame or 60 samples per second. It then matches a phoneme database of the user's previously recorded voice with the real-time sound samples. From this comparison, it returns its best guess as to what phoneme is being used. This is stored as a set of data values depicting the phoneme and the current frame number. When using this method for Sparky, the stored data values were run through a script where they were used to automatically set the phonemes for Sparky's dialogue. Unfortunately, this caused a distracting jitter to occur in Sparky's mouth, even after several steps were taken to blend and smooth it.

There were two main reasons for this. First, the program returned a phoneme value for each frame. If it guessed wrong just ten percent of the time, the result would still be three jitters every second. Second, it didn't take into account the blending or slurring of phonemes as naturally occurs in speech. For example, take two common words "all" and "those". If said slowly and separately, five phoneme shapes result. Two, *ah* and *l* shapes, for "all" and three, *th, oo* and *s* shapes, for "those". However, when said together, specifically if said quickly, the *l* and the *oo* shapes are glossed over and completely lost. The *sound* is still made, but the shape is not. Instead, the correct way to animate this would be with three shapes. Beginning with the *ah* to *th* shapes and ending with the *s* shape.

Even if this method worked flawlessly, jitter would inevitably occur when natural problems like phoneme blending arose in the dialogue. By running a post process "cleanup" program, one could conceivably solve these problems and create an accurate automated lip-synch program. However, this could have easily become a thesis in itself.

Once this was realized, the automated lip-synch system was dropped. Instead it was determined that animating the lip-synch by hand was going to be the quickest process. Unfortunately, two weeks were lost working on the failed automated system, compared to the five days spent animating the final lip-synch.

There are a few additional pointers that should be useful for anyone hoping to do a similar project. First, only nine phonemes are truly necessary. The twelve phonemes used in this project were a bit excessive. For example, the *ah* and *oh_uh* shapes were interchangeable, as were the *ae_eh_ih_es* and *ee_y* shapes, and the *s_z* and *d_g_k_t* shapes. Second, if the character is not modeled in the default *m_b_p* position, then make the *m_b_p* position the default. In other words, turn the attribute driving this phoneme shape fully on and then lock it. Animate all the other phonemes excluding this one. This allows the closed position to become the default position and will save in much confusion during lip-synch animation. Third, exaggerate the phonemes. The face will look much more alive and read much more clearly if the phonemes are exaggerated. Specifically exaggerate the width of the corners of the mouth as they move in and out. Finally, create an additional attribute driving the tongue in and out. This will correct any times the tongue passes through the teeth while blending between phonemes.

IV.3.3. The Eyes

The rigging of the eyes was the second area which cost production time. The loss came not in the rigging process, but later during animation. As stated previously, the eyes were fashioned in a cartoon-like style. This was the root of the resulting animation problems.

In nature, eyes are spheres. This creates a consistent perception of the eye by the viewer. It allows the viewer to understand and follow the gazing direction. Because of the oval shape of Sparky's eyes, problems arose with Sparky's perceived gaze. As Sparky's head would turn, the perception of his gaze would become skewed. This created the need for his eyes to be readjusted far more frequently then if they had been spherically shaped.

This also created a problem with the eyelids. When an eye looks up, the eyelid moves up. Conversely, when the eye looks down the eyelid drops as well. However, due to Sparky's skewed gaze perception and special rigging system, an expression to automate this change could not be accurately made. Therefore, the eyelids had to be adjusted by hand during animation. This lead to another animation pass that could have been avoided.

IV.3.4. Scalability

When it comes to rigging for animation, a good guideline to follow is to always allow for more than enough movement. If a hand should bend forty degrees then it is a good idea to give it the ability to bend ninety degrees. This ensures that no matter what the animator needs for a specific motion it will be possible to achieve. This type of extra movement was built into the rig for Sparky, not only in the form of wider ranges of motion, but as ranges in scale as well. Sparky's arms and hands were given the ability to individually change in scale. This turned out to be immensely useful. It allowed for more visually dynamic poses to be explored and for greater ranges of motion to be used. This was an invaluable addition and should be strongly considered for any rig.

IV.4. Lighting

To save some production time, the lighting was set up before the animation was started. Because the flow of the dialogue was broken into ten "scenes" it made sense to animate each scene separately. This would keep file sizes to a minimum as well as make file and rendering management much less complicated. Therefore, to avoid having to light ten different animated scenes later, the lighting was done once, initially.

IV.5. Shading

The shading was finished before the animation as well. It was done for the same reasons as the lighting, so that it only had to be completed once. Certain steps were taken during shading in order to keep production times minimal. Any textures needed were created procedurally rather then using image based texture maps. This saved on the time to create the texture map images, and it avoided any *UV mapping*. In this manner, UV mapping, a time consuming process in which the user defines where on a texture map each vertex of an object lies, was kept to a bare minimum. In fact, it only had to be done on two objects, the face and the goggles. *Environment maps*, images mapped to a specified object to serve as the reflected color data, were also used. Objects which required reflections like Sparky's eyes and antenna were given these maps. This enabled the time consuming rendering technique of *ray tracing* to be avoided completely.

IV.6. Animating

From the beginning, it was known that the time gained or lost in the animation phase of the production could make or break the project. In fact, almost everything done to assist in the automation of Sparky was designed to quicken this stage of the process. There was no way this production could afford to spend the two years it would take completing the animation at current weekly industry standards.

Fortunately, the animation process for Sparky turned out to be a huge success. It started off fairly slowly with the completion of only forty seconds of animation in the first five weeks. However, animating grew easier as working with the character's controls became more and more familiar. The process also gained momentum as the timing of the character, his mannerisms and movements became more intuitive. Even the steps taken to speed up interaction proved successful. The computer had no noticeable delay during interaction. In all, it took just under five months and a total of 720 hours to complete. At its peak, over 40 seconds of animation was finished in just six days, while an overall average of 16.2 seconds of animation was completed each week. This was more then five times the weekly animation requirement of a major studio. The rig had proven its worth.

IV.6.1. The Trax Editor

The Trax editor provided in Maya [8] also aided in reducing animation times. As previously discussed, the editor allowed "clips" of finished animation to be reused and mixed with current animation. Repeated actions, like Sparky straightening his antenna, were done using these clips. In addition, a few exit sequences were reused. However,

there was an issue which limited the ability to reuse clips. Blending smoothly from clips back to keyed animation was quite difficult. If accurately keyed animation was already set, then there was no problem. However, the animator had to guess where in space the keyed animation would take place relative to the animation contained in the clip. The problem lay in the fact that it was a system of trial and error to create a seamless transition from the clip back to the key framed animation. After working with Sparky for several months, it actually became faster to animate movement similar to that of the clip rather then try to make the clip blend seamlessly.

IV.7. Rendering

The rendering time, per frame, was also kept extremely low. Due to the steps taken in the shading phase, no ray tracing was needed. Motion blur was also avoided, while ambient occlusion was faked. Avoiding these three calculations reduced render times significantly. However, the three render passes illustrated in Fig. 20 were still needed. Fortunately, the combined render time for all three passes was about one minute and twenty seconds per frame. The rendering was completed in less then ten days.

CHAPTER V

CONCLUSION

The work reported in this thesis resulted in more then five minutes of fully animated character monologue. Deadlines and budget constraints were met throughout the production process. Time-saving actions were taken in each production phase, resulting in the animation being done at a rate five times faster then current industry standards [12]. The time saved in the animation phase, as well as in others, allowed the project to finish on time and within budget. The specific strategies used to simplify, streamline, and automate the production process at low cost were recorded and evaluated.

An area of this thesis which could be streamlined farther is the automation of the lip-synch. Time constraints did not allow the use of full lip-synch automation. For the approach taken in this case, a post process "cleanup" program could conceivably solve the previously stated problems and create an accurate automated lip-synch program. However, creating this cleanup program would take some research and innovative blending techniques. A system based on breaking down the written dialogue into blended phonemes might be another more feasible approach. Haaser's [5] program could then be used to obtain the timing data for each phoneme.

Intelligent choices and clever use of automation and streamlining techniques make it possible to complete an educational production involving extensive character animation on a limited budget. This thesis displays such results in its demonstration project, and may be used as an initial reference of cost cutting ideas for anyone who is interested in stretching their own animated production's budget.

REFERENCES

[1] J.F. Blinn, "The Ancient Chinese Art of Chi-Ting." *SIGGRAPH 88 Course Notes*, 1988.

[2] T. Capizzi, *Inspired - 3D Modeling and Texture Mapping*. Boston, MA: Premier Press, 2002.

[3] S. Culhane, *Animation from Script to Screen*. New York: St. Martin's Press, 1988.

[4] S. Culhane, *Talking Animals and Other People.* New York: St. Martin's Press, 1986.

[5] C. Haaser, "Automatic Real-Time Lip Synchronization Using LPC Analysis and Neural Networks." Master's thesis, Texas A&M University, College Station 2002.

[6] J. Halas, *Masters of Animation.* Topsfield, MA: Salem House Publishers, 1987.

[7] P. James, "History of Animation." < http://www.paknet.com.pk/users/nmalik/COMM4-TTA-03/History%20of%20Animation%20Before%20Disney.htm > (accessed 8 December 2003).

[8] J. Kundert-Gibbs and P. Lee, *Mastering Maya 3*. Berkeley, CA: SYBEX Inc., 2001.

[9] J. Lasseter, "Principles of Traditional Animation Applied to 3D Computer Animation." *Proc. ACM SIGGRAPH, SIGGRAPH 87, Computer Graphics, 21(4)*. pp. 35-44. July 1987.

[10] Microsoft Corporation. *MSDN Library: Animation Principles (Agent: Platform SDK)*. 2003. < http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msagent/deschar_8ab6.asp > (accessed 20 November 2003).

[11] Pixar. "Company Info." 2004. < http://www.pixar.com/companyinfo/history/ > (accessed 10 January 2003).

[12] K. Ram, "Pixar Animator Alex Orelle Speaks." June 2002. < http://asifa.net/israel/alexorell-english.html > (accessed 19 April 2004).

[13] C.A. Stablile and M. Harrison, *Prime Time Animation – Television Animation and American Culture.* New York: Routledge, 2003.

[14] M. Wrabel, "Principles of Animation." November 2001. < www.lowrestheater.com/ds3lab/downloads/Principles.pdf > (accessed 10 January 2003).

Supplemental Sources Consulted

V.R. Auzenne, *The Visualization Quest – A History of Computer Animation.* Cranbury, NJ: Associated University Presses, 1994.

C. Caldwell, "Preproduction – The Industry Secret." *Computer Graphics*, pp. 30-31. May 2001.

E.J. Cheetham, "The Nuts and Bolts of Animation." *Computer Graphics*, pp. 48-52. May 2001.

M. Ford and A. Lehman, *Inspired - 3D Character Setup.* Boston, MA: Premier Press, 2002.

J. Hayes, "Creating Character Animation Assets." November 1999. < http://www.gamasutra.com/features/19991105/hayes_02.htm > (accessed 25 June 2004).

E.L. Levitan, *Handbook of Animation Techniques.* New York: Van Nostrand Reinhold Company, 1979.

P. Lord and B. Sibley, *Creating 3-D Animation.* New York: Harry N. Abrams, Incorporated, 1998.

A. Neuwirth, *Makin' Toons – Inside the Most Popular Animated TV Shows and Movies.* New York: Allworth Press, 2003.

D. Sturman, "The State of Computer Animation." *Computer Graphics*, pp. 57-61. February 1998.

F. Thomas, "The Future of Character Animation by Computer." *NCGA 84 Animation Course Notes*, 1984.

R. Williams, *The Animator's Survival Kit.* New York: Faber and Faber, 2001.

VITA

Luke Anthony Carnevale
6671 Nyman Dr.
Dallas, TX 75236
luke@viz.tamu.edu

Education

M.S. in visualization sciences	Texas A&M University, 08/04
B.E.D. in environmental design	Texas A&M University, 05/01

Employment

Texas A&M University, College Station, TX	Head Animator on NSF Show, 09/02-03/04
Texas A&M University, College Station, TX	Graphic Artist, 05/02-09/02
Texas A&M University, College Station, TX	Graduate Assistant, 09/01-01/02
Chili's Bar and Grill, College Station, TX	Waiter, 05/98-08/01